Junior Frontend Developer – Battleship Task

# Goal

The goal of this task is to develop a web application for the classic [Battleship game](https://en.wikipedia.org/wiki/Battleship_(game)). For this task you must use [React.js](https://facebook.github.io/react/). The game logic will be implemented for you via a REST API.

The candidate must complete the task 7 days after receiving this document. We don’t expect the task to take 7 full days of work, we simply want to give a candidate enough time, especially if she/he is working a full-time job.

We will be evaluating the candidate ability to work with an existing API and solving frontend related issues. We will also be closely looking at the quality, “cleanness” and general structure of the code. Thus, the candidate should aim to write the best code that she/he can.

# Requirements

Required

* The game should allow a player to play against the computer
* The UI interface should be divided into 2 parts:
  + A left panel with a list of games (simply show the Game ID)
  + Once a game is selected, the right panel should show the grid in which the game can be played. The grid size is 10x10
* For sake of clarity, this version of the game has only one grid where all the ships are positioned. Computer’s and player’s ship will never overlap.
* There should be a button to start a new game. The API will randomly position the ships for you.
* Clicking on the left panel, I can select any past game to play it. Games can be interrupted at any time. If I want I should be able to play multiple games at once by switching between games using the left panel (i.e. only one game is displayed at the time)
* The player always starts
* The grid will show:
  + The position of the player’s ship in colour A
  + The cell where the player has shot but missed in colour B
  + The cell where the player has shot and hit in colour C
  + The cell where the computer has shot but missed in colour D
  + The cell where the computer has shot and hit the player’s ship in colour E
  + (The colour above are just an indication. If the candidate wishes she/he can use any assets as long as it is easy to differentiate the different cases)
* By clicking on an empty cell, a player will fire a missile and attempt to hit one of the computers ship. It should not be possible to fire a missile where a missile has already been fired
* The API will respond with the computer move after each shot by the player
* A ship is sunk when all its cells have been hit. This should be made visible on the grid so the player knows when to stop shooting.
* The player wins if all the computer’s ships have been sunk
* The computer wins if all the player’s ships haven been sunk
* Once a game is finished the player cannot shoot anymore missile

## Optional

* Ability to delete old games
* Show where the last shot done by the computer (e.g. use different border colour or make it glow)

# API

The candidate is provided with an API which handles all the game logic:

* Creation of game
* Ship placement
* Deciding if a ship hit or miss
* Shooting on behalf of the computer
* Winning conditions

## Security

All the API calls are secured with an API key provided to the candidate.

Every HTTP request must have a `x-api-key` header with value the key given to the candidate

## API details

The battlefield is modelled as a matrix, i.e. an array of array. If a grid was 2x2 a battlefield would look like that:

|  |
| --- |
| [  [0, 0],  [0,0]  ] |

In this task the grid is 10x10 so a battlefield object will have 10 arrays of 10 elements.

Each cell is an integer encoded as the following:

|  |
| --- |
| UNKNOWN = 0  PLAYER\_SHIP = 1  ENEMY\_SHIP = 2  PLAYER\_SHIP\_HIT = 3  ENEMY\_SHIP\_HIT = 4  PLAYER\_MISS = 5  ENEMY\_MISS = 6  PLAYER\_SHIP\_SUNK = 7  ENEMY\_SHIP\_SUNK = 8 |

When referring to a position we use an array like:

|  |
| --- |
| [1, 2] |

The first element of the array is the row number, starting from 0. Thus, this is the index in the first array

The second element of the array is the column number, starting from 0. Thus, this is the index in a sub-array

The base URL is <https://zyqh9s9xt4.execute-api.eu-west-1.amazonaws.com/prod/> all the further requests are relative to this URL

The candidate will be provided with a Swagger JSON file which can be uploaded [here](http://editor.swagger.io/). This should give details about all the API methods, including the JSON schema used for requests and response.

The candidate will also be provided with a Swagger file that can be imported into [Postman](https://www.getpostman.com/). We strongly recommend the candidate to use Postman for testing the API. It is an excellent REST client and it is very easy to use.